Hello welcome back to learning Qt5.

This is Symeon Huang. In this video we’re going to talk about command-line processing.

We're going to take a look at how to add command-line options and how to parse and process command-line options.

So if you're ready. Let's open our previous project HelloWorld. Here as you can see I added a few lines.

I include QCommandLineOption, QCommandLineParser.

I set the application name to QCoreApplication and the application version. In the end it just out old “Hello Qt World” friend and print number out.

What's the difference is I added a parser and the parser will process the QCoreApplication.

As you can see, the QCoreApplication is initialized with all arguments. And by passing “a”

to the parser, we actually also pass the argument an argument list to the QCommandLineParser.

Here we add the command help option and command version option.

Note if you do not set application name operation, you will not see anything when you try to print out

the version and how do we add command-line argument when we run the application.

In Qt creator, it is quite easy.

Switch to projects, here you can add all your command-line arguments. Let’s check our Help first. That would be dash h.

I would just suggest to you to check these checkbox to run your application in the terminal.

So run it. The checkbox will essentially give you a pop-up, instead of running the application, it will open a new

window to show all the results instead of showing all the application outputs in the panel.

As you can see it's just a very simple help information from any command-line options you'd expect. Now the dash h would

actually stop the processing and print out the help information as what would expect.

What about dash v. Would be version.

And here is the information we get.

Hello Word, the application name, and then 0.1, the version.

If we don't give any command-line argument.

the application will run just expected, print out Hello Qt World This is number 1.

Because num has value 1.

Now say I want to change this number depending on what number I give to the command-line.

What you can do. You can add a new QCommandLineOption.

The first one will be, with the first argument as the name.

If it's a single character.

It will be treated as a short name.

If it's not single character, it will be treated as a long name. We will use the third overloaded constructor

with the first argument as a name.

The second description, the third as value name and the last one as the default value.

Do note they are all type of QString.

So our default value will be 1.

And here we do not need to initialize, for reason we're going to change it’s initial

value to zero.

The next step we need to do is to add this option to the parser.

However, this is not finished.

Once you processed the argument, all the values will be stored in the parser. You do have to get the value back.

You do have to get the value out from the parser.

What we can do. we can use the value() member function and we give the option that we want the value.

since the value is a QString.

We can use toInt() member function to convert it into an integer.

Now if we just run it without any argument. Which number will we get? Is it number one or number zero.

Let’s find out.

Aha it’s number one because we give default value to one here.

If we remove this argument.

It gives us 0.

Why would be that?

Because we never set numOption.

Hence the parser’s value.. were actually via empty string.

An empty string to the integer will be zero. Because the conversion failed, hence we get zero.

This is not an intended result.

Hence, we have to use a wrapper.

This is not the intended result. As I can show you,

If I put here minus one. And run the application again. You will get minus one instead of zero. Because the value

is never set. toInt() actually failed.

As you can see number zero.

And how do we avoid this unintended result

We can use isSet() function.

We only get the value once this option is actually set.

Now after these changes we should have be able to get minus one instead of zero failure.

It’s minus one now.

Now I want to give a number myself from the command-line.

Let's switch to projects.

We can use dash n2. But how can i know it’s dash n.

Maybe we should check the Help first.

OK.

Aha, as you can see dash n, so we give n the number. As you can see n is just the name of the command-line option we just assigned.

And “the number argument” is the description that we gave. Number is a value string.

Note the value string can not be duplicated.

It may not mean too much in this stage. But it’s a convention to make a sensible name for value string.

OK let's say I want number three. I would say dash n3.

Since I said dash n3, I should have be able to catch 3 here.

Because numOption we set. The value of the numOption is a string 3.

toInt() should have be able to convert this into integer 3.

And lastly I should have be able to see 3 from the output.

Let’s find it out.

This is number 3.

After this video you should be able to enrich your command- line options with QCommandLineParser and QCommandLineOption.

In the next video, we are going to talk about multi-platform and the portability

which is a key selling point as a cross-platform framework.